CMPS 256: Algorithms and Data Structures

Final Exam

Name: .....

ID: .....

Problem 1. (20 pts) True, False and Justify

Circle T or F for each of the following statements to indicate whether the statement is true or false, respectively. Justify your answers. NO CREDIT will be given to any answer that only states TRUE or FALSE without proper justification.

T F By the Master Theorem, the solution to the recurrence $T(n) = 3T(n/3) + \lg n$ is $\Theta(n \lg n)$.

T F In the *worst case*, Merge-Sort runs in $O(n \lg n)$.

T F Given *any* binary search tree, we can print its elements in sorted order in $O(n)$ time

by performing an Inorder-Tree-Walk.

T F $n \lg^n n = \Omega(n^2)$.

T F Given a set of n elements, one can output in sorted order the k elements following the median in sorted order in time $O(n + k \lg k)$.

T F For hashing an item into a hash table in which collisions are resolved by chaining, the worst-case time is proportional to the load factor of the table.

T F A heap can be constructed from an unordered array of numbers in linear worst-case time.

T F If the depth-first search of a graph G yields no back edges, then the graph G is acyclic.

T F Insertion in a binary search tree is "commutative". That is, inserting x and then y into a binary search tree leaves the same tree as inserting y and then x.

T F A heap with n elements needs at least $O(n \lg n)$ time to be converted into a binary search tree.

Problem 2. (40 pts) True, False and Justify

Circle T or F for each of the following statements to indicate whether the statement is true or false, respectively. Justify your answers. NO CREDIT will be given to any answer that only states TRUE or FALSE without proper justification.

T F The solution to the recurrence $T(n) = 64T(n/4) + 8^{\lg(n)}$ is $\Theta(n^3 \lg n)$.

T F The solution to the recurrence $T(n) = 100T(n/99) + \lg(n!)$ is $\Theta(n \lg n)$.

T F For an undirected graph with integer edge weights and where each vertex is degree 2, a Minimum Spanning Tree (MST) can be found in $O(V)$ time.

T F For any directed acyclic graph, there are always several topological orderings of the vertices.

T F A longest path in a DAG $G = (V, E)$ can be found in $O(V + E)$ time.

T F A graph algorithm with $O(E \lg V)$ running time is asymptotically better than an algorithm with a $O(E \lg E)$ running time for a connected, undirected graph $G(V, E)$.

T F If some of the edge weights in a graph are negative, the shortest path from s to t can be obtained using Dijkstra's algorithm by first adding a large constant C to each edge weight, where C is chosen large enough that every resulting edge weight will be nonnegative.

T F Let $G = (V, E)$ be a weighted graph and let M be a minimum spanning tree of G. The path in M between any pair of vertices v_1 and v_2 must be a shortest path in G.

T F Suppose we have a graph $G = (V, E)$ that is strongly connected. For any depth-first search of G, if all the forward edges of G (with respect to the depth-first forest) are removed from G, the resulting graph is still strongly connected.

T F Given a graph $G = (V, E)$ with distinct costs on edges and a set $S \subseteq V$, let (u, v) be an edge such that (u, v) is the minimum cost edge between any vertex in S and any vertex in $V - S$. Then, the minimum spanning tree of G must include the edge (u, v).

Problem 3. (10 pts) Pattern Matching

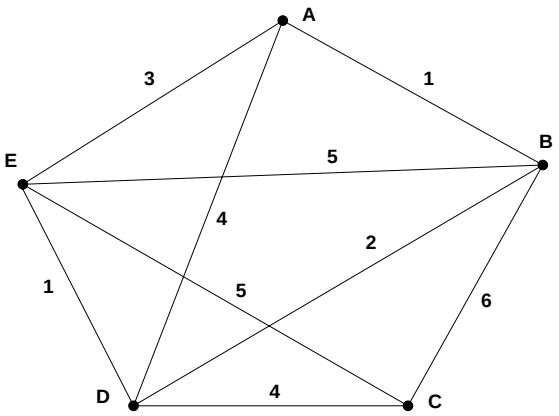
You are given a *text* array $T[1 \ldots n]$ containing characters from the Latin alphabet (*i.e.* $T[i] \in \{a, b, \ldots z\}$ for $i = 1 \ldots n$. Moreover, you are given another array $P[1 \ldots m], m < n$, which also contains characters from the Latin alphabet. For any sub-array $T_i^m = [i \ldots i + m - 1]$ of T, we say that T_i^m is a pattern match of P if there is a way of permuting symbols in T_i^m so that the resulting array is equal to P.

Give what you think is the most efficient algorithm that, given indices i and m, determines if T_i^m is a pattern match to P.

Problem 4. (10 pts) Modifying MST

Consider the weighted graph below having a unique Minimum Spanning Tree (MST).

1. Draw the Minimum Spanning Tree T of this graph.
2. If we add the edge AC to the graph, the new graph may have the same MST, a new MST, or both. Explain how you can tell which case applies by examining the weighted graph $T \cup \{AC\}$.
3. Which case applies if AC has weight 5?
4. Which case applies if AC has weight 4?
5. Which case applies if AC has weight 3?



Problem 5. (20 pts) Directed Acyclic Graphs

a. Consider a directed acyclic graph (DAG) $G = (V, E)$. A topological sort of the vertices V of the graph is a linear ordering of the set V so that if an edge (u, v) exists in G, vertex u appears before v in the order. It is possible to generate a topological ordering of the vertices of a graph from the information generated by the depth first traversal algorithm. Write pseudo-code for topological sort, and analyze the runtime of your algorithm.

b. Give, in pseudo-code, an efficient algorithm for counting the total number of paths in a directed acyclic graph starting from a given vertex s. Your algorithm should generate and store, for every vertex v, the number of paths from s to v. This number should be stored at the vertices. Analyze your algorithm. *Hint: Use Topological Sort.*